Image & Video Processing

Assignment 3 - Frequency Domain Processing

Albert Cerfeda, Alessandro Gobbetti

Contents

15 May 2023
Università della Svizzera italiana
Faculty of Informatics
Switzerland

1 Theoretical exercise [2 points]

In this section, we will show that if $f(x,y) = g(x) \cdot g(y)$, then the Fourier transform of $F(u,v) = G(u) \cdot G(v)$, where $G(u)$ is the Fourier transform of $g(x)$.
We know that a 2D Fourier transform is defined as:

$$F(u,v) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x,y) \cdot e^{-j2\pi(ux+vy)} dxdy$$

We can substitute $f(x,y)$ with $g(x) \cdot g(y)$:

$$F(u,v) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} g(x) \cdot g(y) \cdot e^{-j2\pi(ux+vy)} dxdy$$

We can split the integral in two parts:

$$F(u,v) = \int_{-\infty}^{\infty} g(x) \cdot e^{-j2\pi ux} dx \cdot \int_{-\infty}^{\infty} g(y) \cdot e^{-j2\pi vy} dy$$

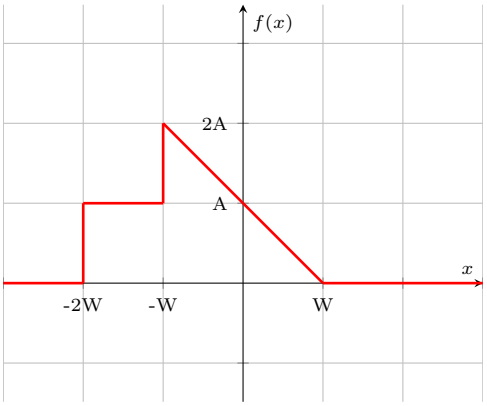
We can recognize the Fourier transform of $g(x)$ and $g(y)$:

$$F(u,v) = G(u) \cdot G(v)$$

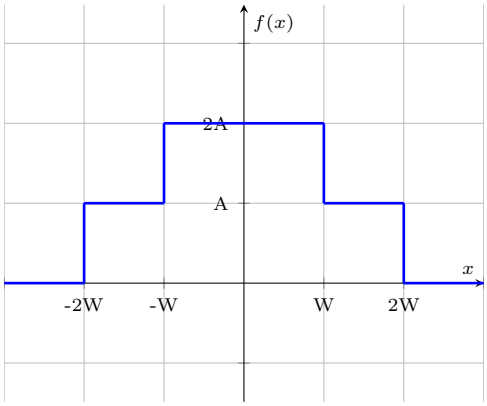
Therefore we have shown that if $f(x,y) = g(x) \cdot g(y)$, then the Fourier transform of $F(u,v) = G(u) \cdot G(v)$.

2 Theoretical exercise [2 points]

We compute Fourier transform $G(\mu)$ of function $g(x) = f(x) + f(-x)$, where $f(x)$ is the function plotted in Figure 1a. The function $g(x)$ is plotted in Figure 1b.



Spatial domain function $f(x)$.

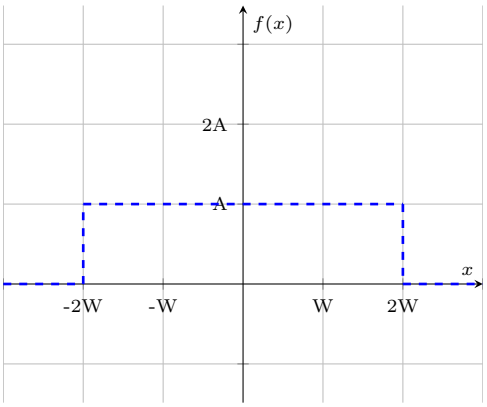


Spatial domain function $g(x) = f(x) + f(-x)$.

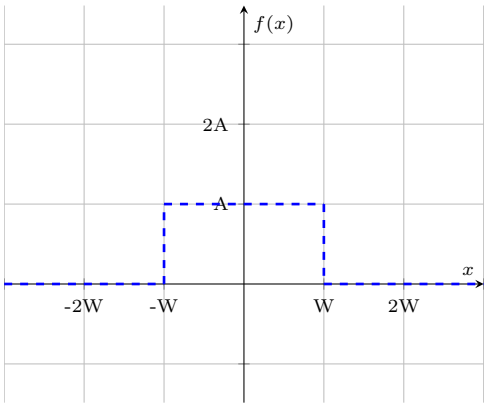
Figure 1: Spatial domain function $f(x)$ [red] and $g(x) = f(x) + f(-x)$ [blue].

By looking at Figure 1b, it can be seen that the function $g(x)$ can be represented as the sum of two box functions. Figure 2 shows the two box functions $h_1(x)$ and $h_2(x)$ that sum up to $g(x)$.

The Fourier transform of a box function $h(x)$ is known: $H(\mu) = AW\text{sinc}(\mu W)$, where A is the amplitude of the box function and W is the width of the box.



Box function $h_1(x)$.



Box function $h_2(x)$.

Figure 2: The two box functions that sum up to $g(x)$.

We can thus express the Fourier transform of $g(x)$ as the sum of the Fourier transform of $h_1(x)$ and $h_2(x)$:

$$\begin{aligned} G(\mu) &= H_1(\mu) + H_2(\mu) \\ &= A4W\text{sinc}(\mu 4W) + A2W\text{sinc}(\mu 2W) \\ &= A2W\left(2\text{sinc}(\mu 4W) + \text{sinc}(\mu 2W)\right) \end{aligned}$$

3 Theoretical exercise

Reducing the size of an image can lead to aliasing unless a low-pass filter is applied before. The goal of this exercise is to find the smallest standard deviation of a Gaussian filter that can be applied in the spatial domain to avoid aliasing when reducing the size of an image by a factor of c.

When sampling, we make use of the *Impulse signal* to take equally-spaced samples. The sample density is defined through the constant ΔT. Reducing the size of an image by a factor of c means that when performing downsampling we sample one pixel every c pixels, i.e. $\Delta T = c$. It is important to note that the Fourier transform of an Impulse signal $s_{\Delta T}(t)$ with $\Delta T = c$ in the spatial domain results in another Impulse signal defined in the frequency domain $\mathscr{F}\{s_{\Delta T}(t)\}$with $\Delta T = \frac{1}{c}$. The translation from the spatial domain to the frequency (Fourier) domain can be visualized in Figure 3:

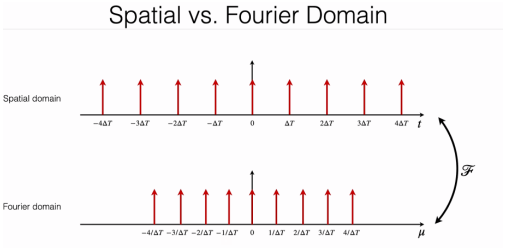


Figure 3: Equivalent Impulse signals in spatial and frequency domain

Convoluting the Fourier transformations of the image with an Impulse signal results in the image spectrum being repeated for every Impulse point in the Impulse train. If the Impulse train is too dense (i.e. ΔT is relatively small) the repeated image spectrums may overlap in certain positions. This leads to aliasing

artifacts. This phenomenon is visualized in Figure 4.

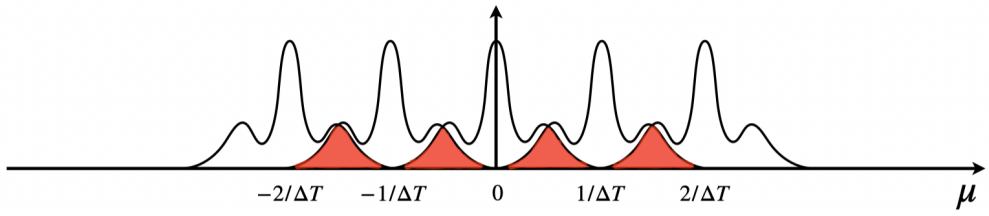


Figure 4: $\mathscr{F}\{s_{\Delta T}(t)\} \star \mathscr{F}\{f(t)\}$

As a countermeasure we apply Gaussian filtering on the image to get rid of high frequencies, resulting in the image spectrum being "narrower" and able to fit in the resulting spectrum without incurring overlaps. We should thus find a Gaussian kernel that does not overlap in the frequency domain as shown in Figure 5.

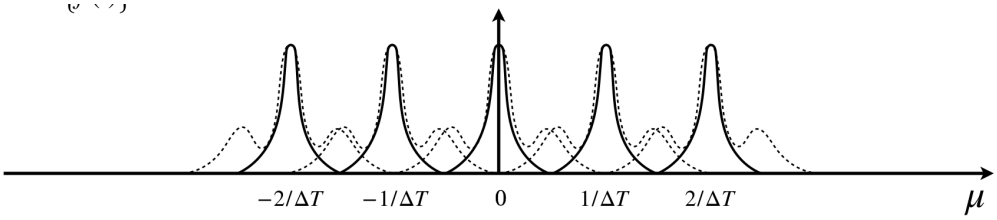


Figure 5: $\mathscr{F}\{s_{\Delta T}(t)\} \star \mathscr{F}\{f(t) \star g(t)\}$

Given that $\Delta T = c$ is in the spatial domain, it is $\Delta T = \frac{1}{c}$ in the Fourier domain. We can use the size of the Gaussian kernel when performing prefiltering on the image to impose a bound on the width of the image spectrum in order not to incur into overlaps.

However, a little bit of overlap is not a problem, for this exercise, we can assume that the spatial frequencies which are attenuated by a filter with a factor smaller than 0.75 do not cause aliasing. We thus need to find the standard deviation σ of the Gaussian kernel in the Fourier domain such that all the values are attenuated by a factor of at least 0.75.

From the Gaussian function:

$$G(x) = e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2},$$

where μ is the mean and σ is the standard deviation. We want then to find that σ such that a Gaussian

function with mean 0 and standard deviation σ is attenuated by a factor of 0.75 at $x = \frac{1}{2c}$:

$$\begin{aligned}
0.75 = G(x) &= e^{-\frac{1}{2}(\frac{x-0}{\sigma})^2} \\
0.75 &= e^{-\frac{1}{2}(\frac{x}{\sigma})^2} \\
\ln(0.75) &= -\frac{1}{2}(\frac{x}{\sigma})^2 \\
\sigma^2 &= -\frac{x^2}{2\ln(0.75)} \\
\sigma^2 &= -\frac{(\frac{1}{2c})^2}{2\ln(0.75)} \\
\sigma &= \sqrt{-\frac{1}{8\ln(0.75)c^2}} \\
\sigma &= \frac{1}{c}\sqrt{-\frac{1}{8\ln(0.75)}} \\
\sigma &\approx \frac{0.66}{c}
\end{aligned}$$

The standard deviation σ found above is in the Fourier domain. To find the standard deviation σ_s of the corresponding Gaussian kernel in the spatial domain (since the Fourier transform of a Gaussian function is another Gaussian function):

$$\begin{aligned}
\sigma_s &= \frac{1}{2\sigma\pi} \\
\sigma_s &\approx \frac{1}{2\frac{0.66}{c}\pi} = \frac{c}{1.32\pi}
\end{aligned}$$

3.1 Bonus

The Fourier transform of a Gaussian function $g(x) = e^{-x^2}$ is another Gaussian function.

$$\begin{aligned}
\mathscr{F}_x[g(x)](k) &= \int_{-\infty}^{+\infty} g(x)e^{-2\pi kix}dx \\
&= \int_{-\infty}^{+\infty} e^{-x^2}e^{-2\pi kix}dx \\
&= \int_{-\infty}^{+\infty} e^{-x^2}[\cos(2\pi kx) - i\sin(2\pi kx)]dx \\
&= \int_{-\infty}^{+\infty} e^{-x^2}\cos(2\pi kx)dx - i\int_{-\infty}^{+\infty} e^{-x^2}\sin(2\pi kx)dx
\end{aligned}$$

The integral of the imaginary part is zero since the integrand is an odd function and the integration interval is symmetric. We are left with the integral of the real part:

$$\begin{aligned}
\mathscr{F}_x[g(x)](k) &= \int_{-\infty}^{+\infty} e^{-x^2}\cos(2\pi kx)dx \\
&= \sqrt{\pi}e^{-\pi^2k^2}
\end{aligned}$$

which is still a Gaussian function. So the Fourier transform of a Gaussian function is another Gaussian function.

4 Theoretical exercise

Figure 6 shows the magnitude of the Fourier spectrum of a spatial signal $f(x)$, which is two boxes centered around the frequency ω_0. We want to find the spatial signal $f(x)$ assuming arbitrary boxes' width W and height A.

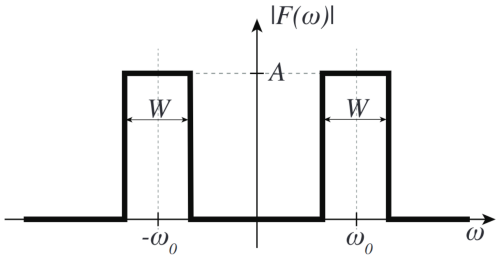


Figure 6: Magnitude of the Fourier spectrum of a spatial signal $f(x)$.

Looking at Figure 6 we can identify multiple rect functions centered around the frequencies $k\omega_0$. We can thus express the Fourier transform $F(x)$ as the convolution of a rect function $rect(x)$ and a sampling function $s_{\Delta T}(x)$:

$$F(x) = AW\texttt{rect}(Wx) \star s_{\omega_0}(x)$$

From the convolution theorem, we know that the Fourier transform of a convolution of two signals is a product of their Fourier transforms and vice versa. We also know that the Fourier transform of a sinc function is a box function and that the Fourier transform of a sampling function is still a sampling function with a sampling period $\Delta T_f = \frac{1}{\Delta T_s}$.

The original signal $f(x)$ is then:

$$f(x) = AW\texttt{sinc}(WX) \cdot s_{\frac{1}{\omega_0}}(x)$$

5 Theoretical exercise

Consider a task of filtering a signal with a spatial domain Gaussian filter $h(x)$. The filter has been approximated with a kernel of inappropriate size (s), effectively windowing the original filter to $g(x)$.

The new filter has now a different Fourier transform than the original one ($G(\omega) \neq H(\omega)$), but it can be expressed as:

$$G(\omega) = H(\omega) \star X(\omega)$$

Our goal is to derive an expression for $X(\omega)$, where s is an additional parameter.

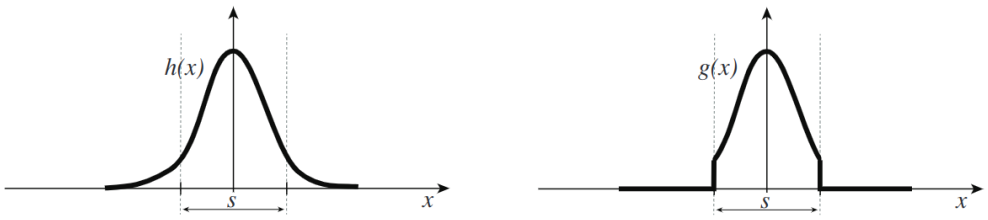
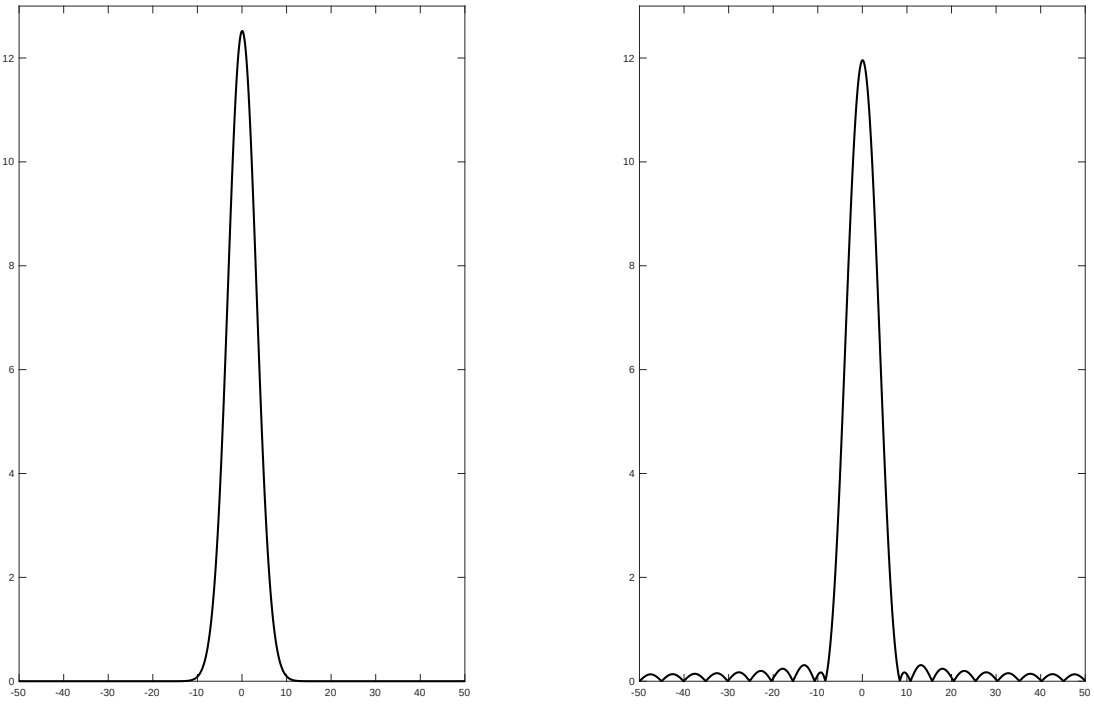


Figure 7: Visualization of the Gaussian filter $h(x)$ and its windowed version $g(x)$.

Looking at the windowed filter in Figure 7, we can express $g(x)$ as the multiplication of the original filter and a box function of width s. The Fourier transform of a box function is a sinc function, therefore, recalling the fact that the product of two signals in the spatial domain is a convolution in the frequency domain, we can express $G(\omega)$ as:

$$G(\omega) = H(\omega) \star (s \cdot \texttt{sinc}(\omega s))$$



$H(\omega)$ $G(\omega)$

Figure 8: Visualization of the Fourier transform of the Gaussian filter $H(\omega)$ and its windowed version $G(\omega)$.

In Figure 8 we can see the Fourier transform of the original Gaussian filter $H(\omega)$ and its windowed version $G(\omega)$. It can be seen that in $G(\omega)$ the convolution with the sinc function has the effect of amplifying some frequencies at the sides which were almost zero in the Gaussian version $H(\omega)$.

6 Gaussian Filtering [4 points]

6.1 Equivalence of spatial and frequency domain filtering

In this exercise we have to implement Gaussian filtering both in the spatial and frequency domain. Namely, we want to show that convoluting a gaussian filter with σ_s in the spatial domain is equivalent to performing point-wise multiplication in the fourier domain between the image spectrum and another gaussian filter with $\sigma_f = \frac{1}{2\sigma_s\pi}$.

- **Spatial domain**

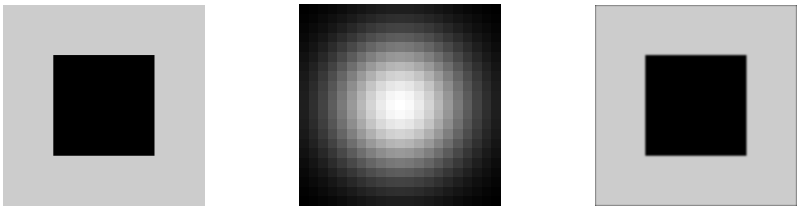
```
function [img_filtered] = spatialGaussianFiltering(img, sigma_s)
        kernel_size_s = 4*sigma_s+1;
        gaussian_filter_spatial = fspecial('gaussian', kernel_size_s, sigma_s);
        img_filtered = conv2(img,gaussian_filter_spatial,'same');
end
```

Results are shown in Figure 9.

- **Frequency domain**

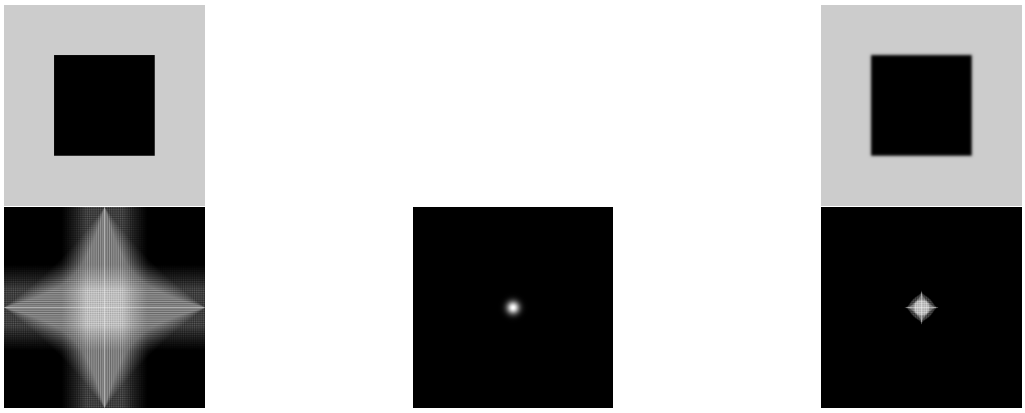
```
function img2_filtered = frequencyGaussianFiltering(img, sigma_s)
    %%% Convert to frequency domain
    img2_fft = fft2(img);
    % Shift lower frequencies to the center of fourier spectrum
    img2_fft = fftshift(img2_fft);

    % Computing gaussian filter
    sigma_f = (1/(2*pi*sigma_s));
    [X,Y]= meshgrid(-size(img,1)/2:size(img,1)/2-1,-size(img,2)/2:size(img,2)/2-1);
    D = sqrt(X.^2 + Y.^2);
    D = D/max(D(:));
    D = exp(-D.^2/2/sigma_f^2);
    D = D/max(D(:));
    gaussian_filter_frequency = D;

    % Perform gaussian filtering in frequency domain
    img2_filtered_fft = img2_fft .* gaussian_filter_frequency;
    % Bring image back to spatial domain
    img2_filtered = ifft2(ifftshift(img2_filtered_fft));
end
```

Results are shown in Figure 10.



Original image Spatial gaussian filter Filtered image

Figure 9: Gaussian filtering pipeline in the spatial domain



Original shifted image Gaussian filter for frequency domain Filtered image

Figure 10: Gaussian filtering pipeline in the frequency domain

6.2 [BONUS] Benchmarking filtering pipelines in the spatial and frequency domain

Let us now compare the two filtering pipelines in terms of execution time. We will vary the spatial sigma σ_s and measure the execution time of both pipelines. Results are shown in Figure 11. We can see that the frequency domain filtering pipeline is faster than the spatial domain one for the majority of values of σ_s.

There are a couple of important remarks to make. We notice how the computational complexity of filtering in the frequency domain comes from computing the fourier spectrum of the image, and therefore depends on the size of the image, while the computational complexity of filtering in the spatial domain depends on the size of the kernel (and therefore on σ in the case of gaussian filtering).
Does this mean that for larger images spatial filtering is to be preferred ? Not necessarily. If an image has to be filtered multiple times (e.g in a image editing software), frequency filtering would require to compute the fourier spectrum only once, amortizing the computational cost over multiple filtering operations. Otherwise, when performing a single filtering operation on a very large image, spatial filtering would be preferred.

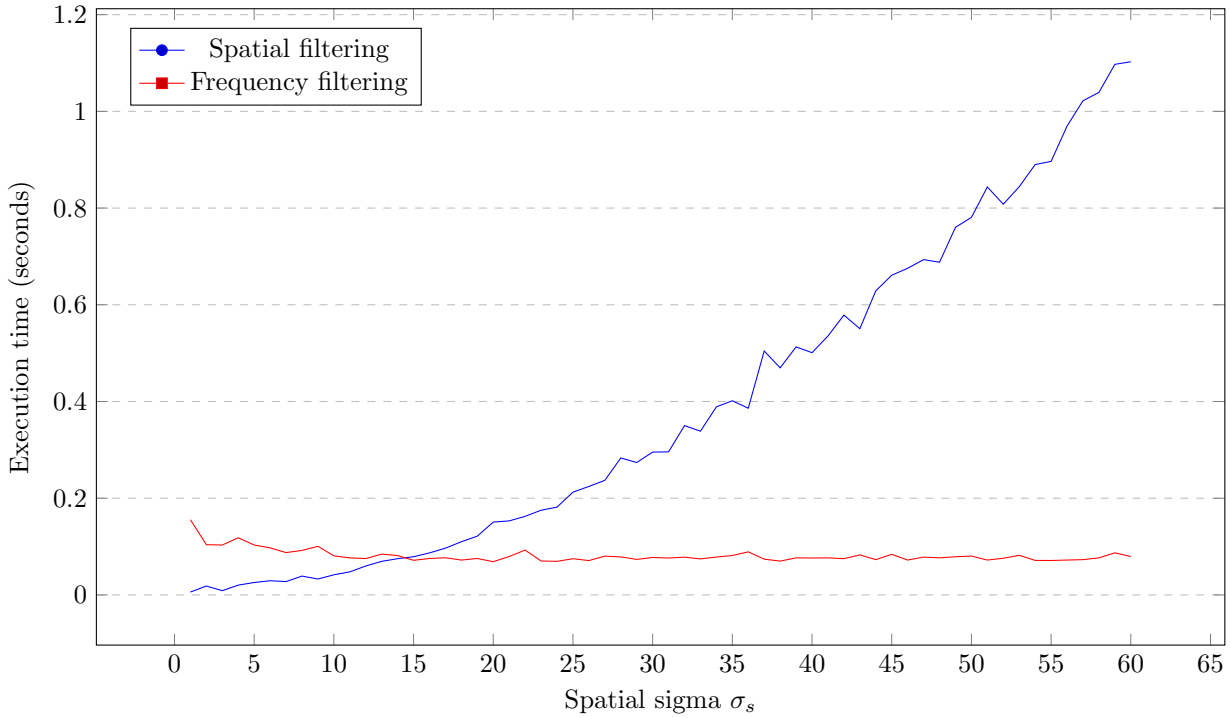


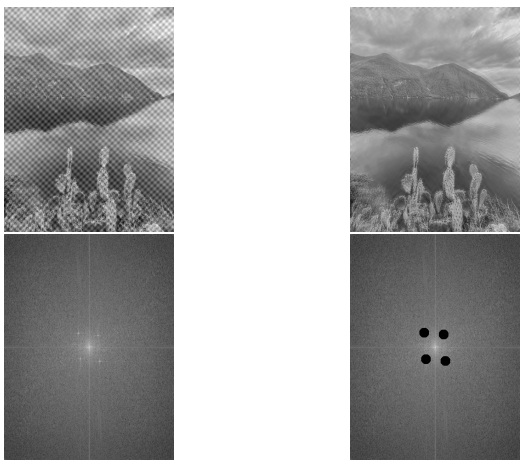
Figure 11: Benchmark comparing Gaussian filtering in the spatial domain vs in the frequency domain

7 Image Restoration [4 points]

We are tasked in removing the checkerboard pattern from the provided image by cancelling out some frequencies in the fourier domain.

```
% ...
% Bring the image in the frequency domain
fft_img = fftshift(fft2(img));

% Compute the mask
mask = circleMask(size(img), 14, [210, 280]);
mask = mask + circleMask(size(img), 14, [265, 285]);
mask = mask + circleMask(size(img), 14, [215, 355]);
mask = mask + circleMask(size(img), 14, [270, 360]);
mask = ~mask;

% Apply the mask
fft_masked = fft_img .* mask;

% Unshift and bring in the spatial domain
fft_res = ifftshift(fft_masked);
img_res = ifft2(fft_res);
function mask = circleMask(imgsize, radius, position)
    [W,H] = meshgrid(1:imgsize(2),1:imgsize(1));
    mask = (((W-position(1)).^2 + (H-position(2)).^2) < radius^2);
end
```

We notice how the checkerboard pattern is oriented diagonally. That is the clear indicator of a sum of equal sinusoids in both vertical and horizontal direction. In fact, the fourier spectrum of the image shows two peaks in both the horizontal and vertical direction. We can therefore cancel the pattern by masking the fourier spectrum in the corresponding frequencies. Results are shown in Figure 12.

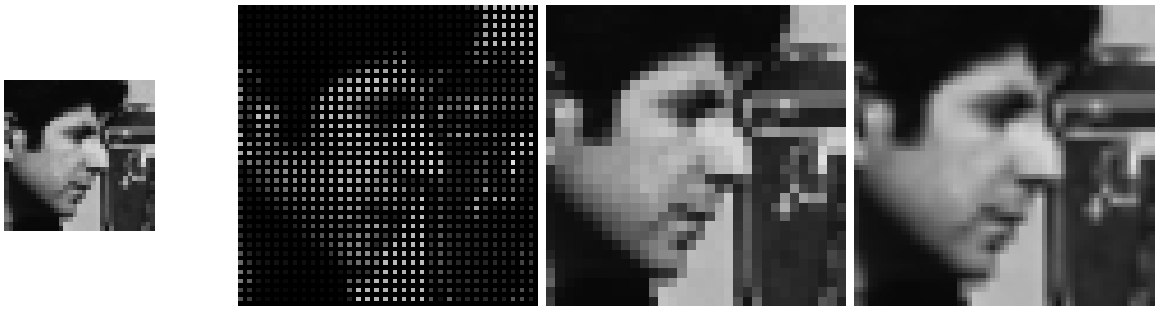


Original image Image with masked pattern

Figure 12: Comparison of different histogram equalization techniques.

8 Bonus

Suppose an image is to be up-scaled by a factor of 2. Firstly, the image is up-sampled as shown in Figure 13b. The missing pixels can then be computed using two different interpolation methods: nearest neighbor (Figure 13c) and linear interpolation (Figure 13d).



Original image | Up-sampled image | Nearest neighbor | Linear interpolation

Figure 13: Up-scaling an image by a factor of 2.

The nearest neighbor method simply copies the value of the nearest pixel to the missing one. The method works by convolving the image with the following kernel:

$$\begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Because of how the up-sampled image is constructed, if the pixel in the center of the kernel is present, the surrounding pixels will be missing and thus the resulting pixel will be the same as the center one. If instead the center pixel is missing, only one of the surrounding pixels will be present and thus used as the value of the missing pixel.

The linear interpolation method instead computes the value of the missing pixel as the weighted average of the four nearest pixels. The method works by convolving the image with the following kernel:

$$\frac{1}{4}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

If the pixel in the center of the kernel is present, the resulting pixel will be the same as the center one. If instead the center pixel is missing, the resulting pixel will be the weighted average of the four surrounding existing pixels in the corners or the mean of the two pixels at the sides if the corner pixels are missing as well.

Figure 14 shows the signal of a single row of the up-scaled image with the two methods. Because of linear interpolation, the linearly interpolated signal is the same as the original one, just stretched by a factor of 2. The nearest neighbor method instead results in a more noisy signal since the missing pixels are simply copied from the nearest existing ones, while in linear interpolation the pixel intensities follow a smoother curve.
In the Fourier domain (Figure 15) this results in the image up-scaled with nearest neighbor interpolation having more high frequencies while the image up-scaled with linear interpolation has the same frequencies as the original image. Thus, the ratio between the Fourier transforms of the linear interpolation up-scaled image and the nearest-neighbor up-scaled image will be close to 1 on lower frequencies (i.e the spectrums are almost identical) and will decrease for higher frequencies (i.e. the spectrum of the linearly interpolated image will have more energy on higher frequencies). This is confirmed by the plot in Figure 15. Intuitively,

it can be seen from the upscaled images that the linearly interpolated image is smoother than the nearest neighbor one, thus having less high frequencies.

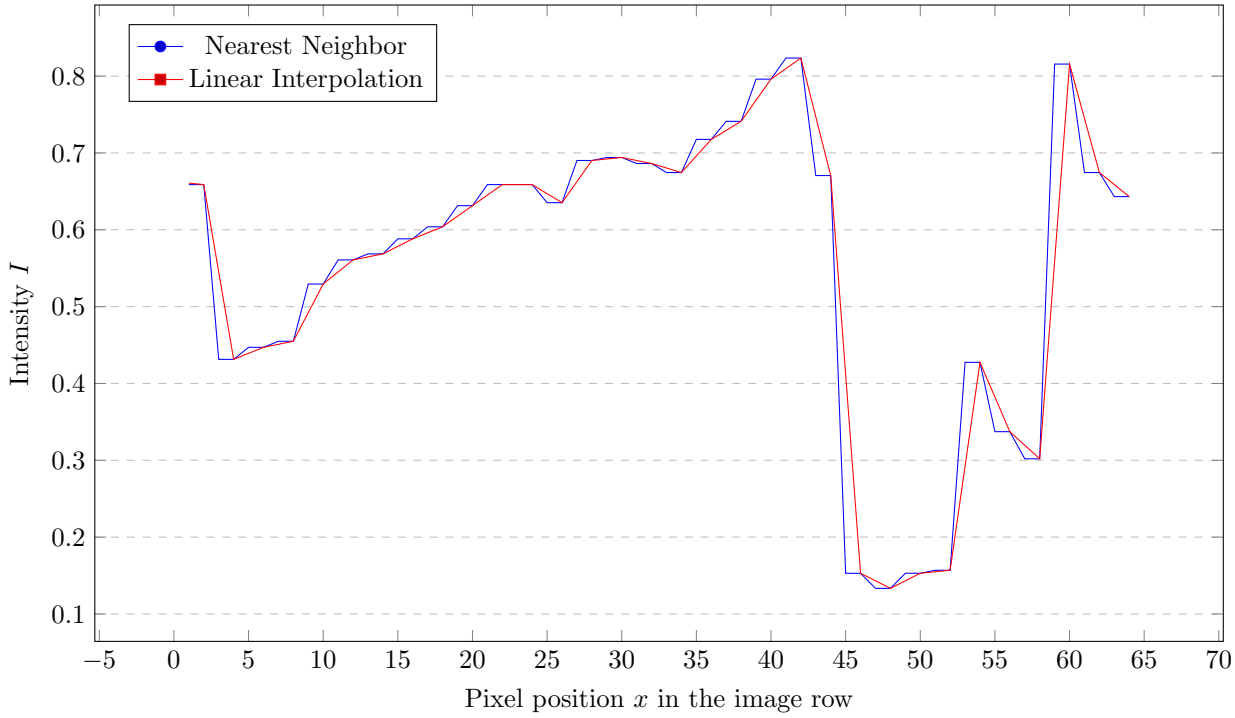


Figure 14: Pixel intensities for one image row of the two interpolation methods.

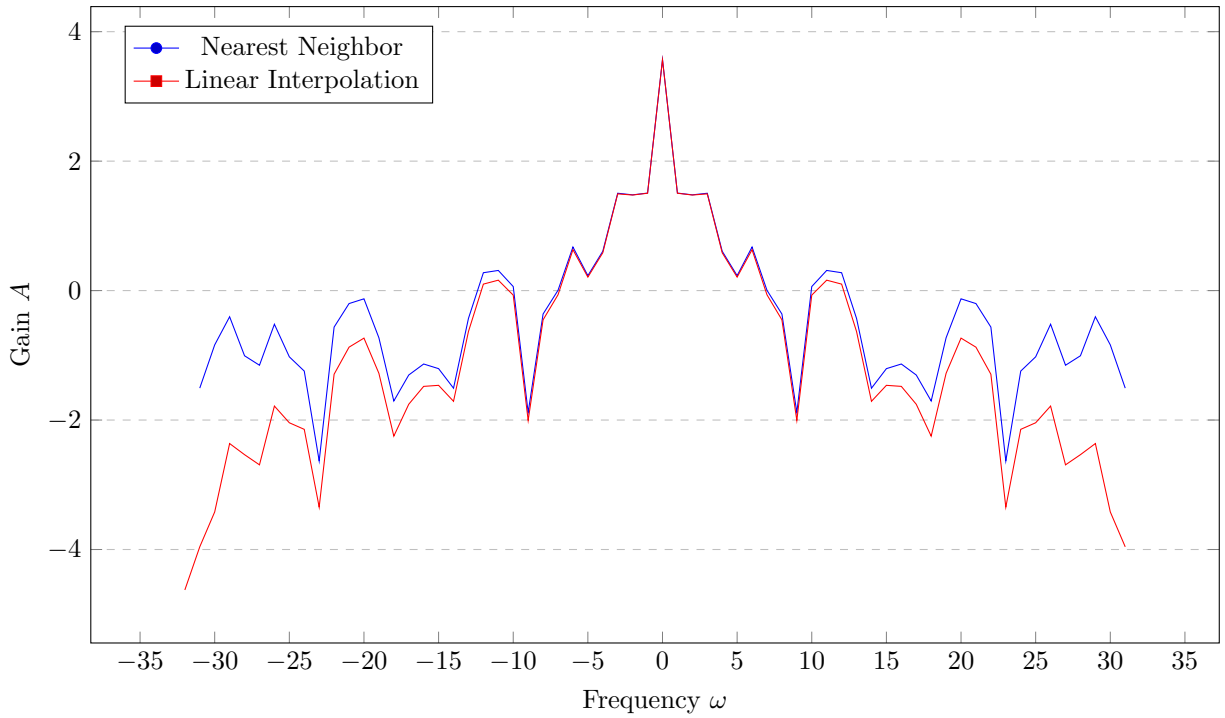


Figure 15: Fourier plot of the two interpolation methods.